

Prácticas Fundamentos de Redes

Chat

Rubén Morales Pérez
Francisco Javier Morales Piqueras

4 de diciembre de 2016

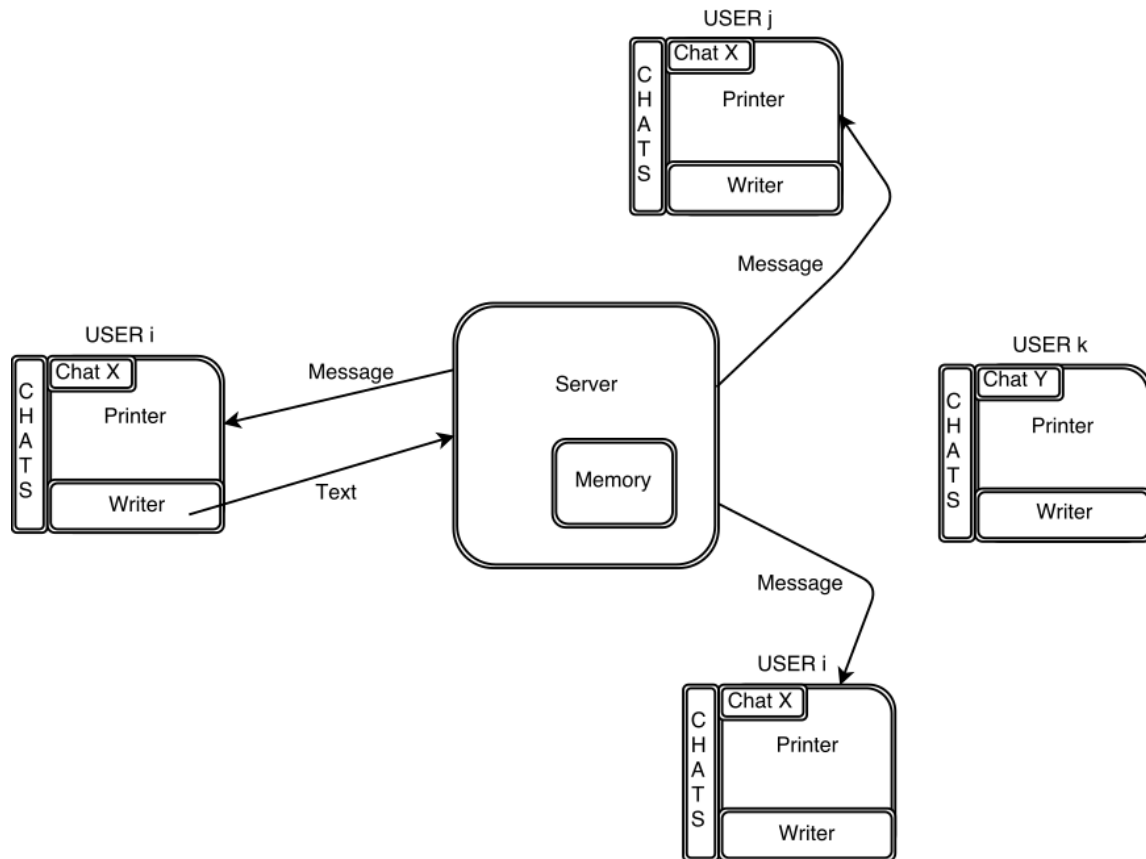
Índice

1. Introducción	2
2. Servidor	2
3. Usuario	3
3.1. Representación	3
3.2. Mensajes	3
4. Configuración	4
5. Funcionamiento	4
5.1. Cliente	5
5.2. Escribir mensajes	7

1. Introducción

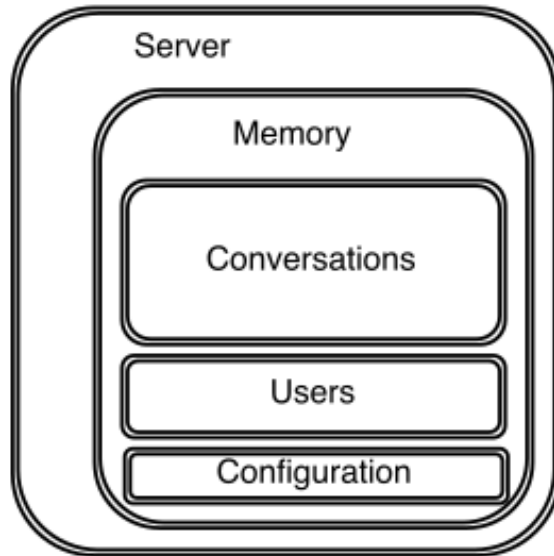
En esta práctica crearemos un chat multiusuario usando el paradigma cliente-servidor. El servidor será un proceso alojado en alguno de los ordenadores de la comunicación. El servidor será el encargado de mandar el mensaje a los usuarios implicados en la comunicación, y solamente a ellos. La práctica será realizada en *JAVA*.

La estructura del programa es la siguiente.



2. Servidor

El servidor será una clase **Server** que estará continuamente recibiendo y mandando mensajes. Guardará los sockets necesarios para la comunicación. El servidor tiene una serie de archivos guardados para su organización.



3. Usuario

3.1. Representación

Un usuario quedará identificado por su *idUser*, los usuarios se guardarán en una carpeta especial con ficheros como el siguiente.

```
1 0
2 ruben
3 password1
4 0
```

Listing 1: 0.user

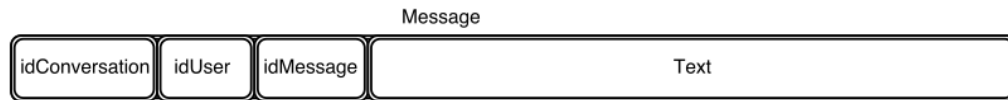
La estructura del fichero es la siguiente

- Identificador de usuario
- Nombre de usuario
- Contraseña
- Grupos a los que pertenece

Esta configuración permite tener un chat con nosotros mismos. Existirán funciones para cargar y guardar usuarios en ficheros.

3.2. Mensajes

Cada usuario tiene una instancia de la clase *Printer* que recibirá y mostrará los mensajes. Los mensajes tienen un identificador del grupo al que pertenecen, del usuario que los envía y un identificador del mensaje. Estos tres parámetros nos permitirán identificar de forma única cada mensaje.



Un fichero de conversación será una sucesión de mensajes, como muestra el siguiente ejemplo.

```

1 0 0 0 20/11/2016 at 06:42:01 - "Message 1"
2 0 1 1 20/11/2016 at 06:42:01 - "Message 2"
3 0 2 2 20/11/2016 at 06:42:01 - "Message 3"
4 0 3 3 20/11/2016 at 06:42:01 - "Message 4"
5 0 1 4 20/11/2016 at 06:42:01 - "Message 5"
6 0 3 5 20/11/2016 at 06:42:01 - "Message 6"
7 0 2 6 20/11/2016 at 06:42:01 - "Message 7"
8 0 1 7 20/11/2016 at 06:42:01 - "Message 8"
9 0 1 8 20/11/2016 at 06:42:01 - "Message 9"
10 0 4 9 20/11/2016 at 06:42:01 - "Message 10"

```

Listing 2: example.chat

Los números al inicio son el identificador de conversación, el del usuario que manda el mensaje y el identificador de su mensaje. Estos tres números en conjunto formarán el identificador de cada mensaje.

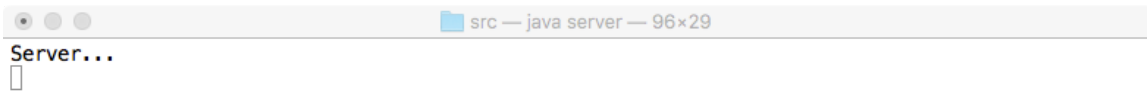
4. Configuración

La configuración del chat la manejaremos con una clase llamada *Config*. En la clase tendremos los directorios con ruta absoluta de las diferentes carpetas necesarias. Habrá variables para controlar cuántos usuarios y conversaciones tenemos registrados en el sistema, y los puertos usados para las comunicaciones. También tendremos guardadas las extensiones con las que guardaremos los ficheros de usuarios y conversaciones.

Al inicializar una instancia de la clase **Config** se crearán las carpetas necesarias para el manejo de usuarios y conversaciones. Tendremos funciones para guardar la configuración actual y para acceder al fichero de cada conversación o usuario. También podremos comprobar si existe un usuario o conversación.

5. Funcionamiento

La interfaz del servidor es sencilla.



5.1. Cliente

Cada usuario tiene un terminal para el Writer y otro para el Printer. ¡

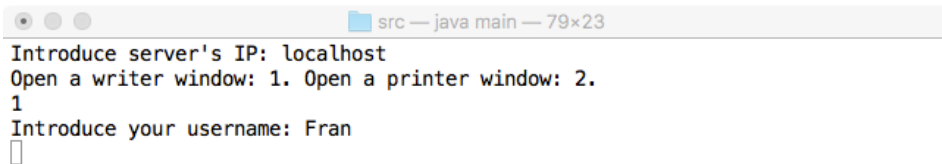
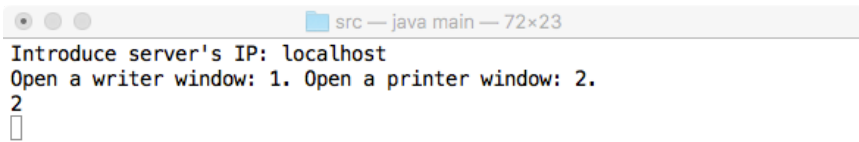


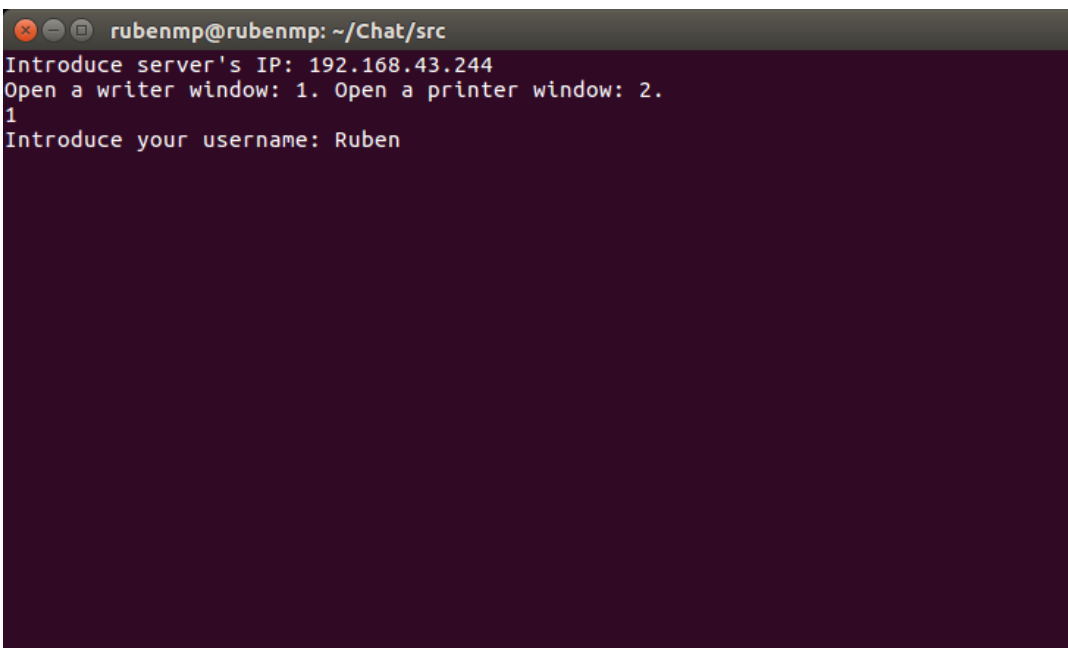
Figura 1: Fran writer

A terminal window with a title bar that says "src — java main — 72x23". The window contains the following text: "Introduce server's IP: localhost", "Open a writer window: 1. Open a printer window: 2.", "2", and a cursor on the next line.

```
src — java main — 72x23
Introduce server's IP: localhost
Open a writer window: 1. Open a printer window: 2.
2

```

Figura 2: Fran printer

A terminal window with a title bar that says "rubenmp@rubenmp: ~/Chat/src". The window contains the following text: "Introduce server's IP: 192.168.43.244", "Open a writer window: 1. Open a printer window: 2.", "1", "Introduce your username: Ruben", and a cursor on the next line.

```
rubenmp@rubenmp: ~/Chat/src
Introduce server's IP: 192.168.43.244
Open a writer window: 1. Open a printer window: 2.
1
Introduce your username: Ruben

```

Figura 3: Ruben writer

```
rubenmp@rubenmp: ~/Chat/src
Introduce server's IP: 192.168.43.244
Open a writer window: 1. Open a printer window: 2.
2
```

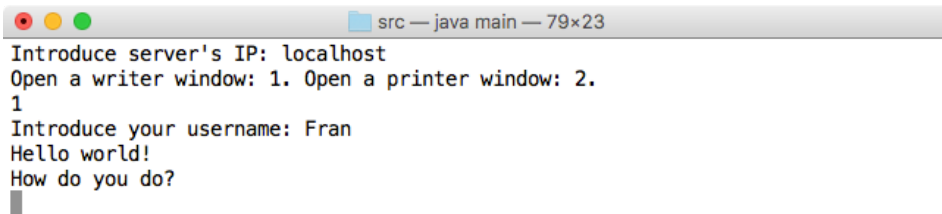
Figura 4: Ruben printer

5.2. Escribir mensajes

Ahora procedemos a escribir mensajes desde ambos ordenadores.

```
rubenmp@rubenmp: ~/Chat/src
Introduce server's IP: 192.168.43.244
Open a writer window: 1. Open a printer window: 2.
1
Introduce your username: Ruben
Hello back!
Fine :)
```

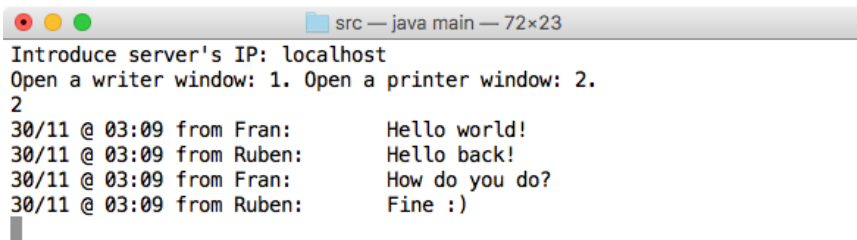
Figura 5: Ruben escribiendo

A terminal window titled 'src — java main — 79x23' with standard macOS window controls. The text inside shows a sequence of inputs: 'localhost', '1', 'Fran', 'Hello world!', and 'How do you do?'.

```
src — java main — 79x23
Introduce server's IP: localhost
Open a writer window: 1. Open a printer window: 2.
1
Introduce your username: Fran
Hello world!
How do you do?
```

Figura 6: Fran escribiendo

Comprobamos que funciona

A terminal window titled 'src — java main — 72x23' showing the output of the program. It includes timestamps and identifies the sender (Fran or Ruben) for each message.

```
src — java main — 72x23
Introduce server's IP: localhost
Open a writer window: 1. Open a printer window: 2.
2
30/11 @ 03:09 from Fran:      Hello world!
30/11 @ 03:09 from Ruben:    Hello back!
30/11 @ 03:09 from Fran:    How do you do?
30/11 @ 03:09 from Ruben:    Fine :)
```

Figura 7: Fran recibiendo


```
rubenmp@rubenmp: ~/Chat/src
Introduce server's IP: 192.168.43.244
Open a writer window: 1. Open a printer window: 2.
2
30/11 @ 03:09 from Fran:      Hello world!
30/11 @ 03:09 from Ruben:    Hello back!
30/11 @ 03:09 from Fran:    How do you do?
30/11 @ 03:09 from Ruben:    Fine :)
```

Figura 8: Ruben escribiendo